Using Virtual Markets to Program Global Behavior in Sensor Networks

Using Virtual Markets to Program Global Behavior in Sensor Networks

Geoff Mainland, Laura Kang, Sebastien Lahaie, David C. Parkes, and Matt Welsh
Division of Engineering and Applied Sciences
Harvard University
{mainland,kang,slahaie,parkes,mdw}@eecs.harvard.edu

Abstract

This paper presents *market-based macroprogramming (MBM)*, a new paradigm for achieving globally efficient behavior in sensor networks. Rather than programming the individual, low-level behaviors of sensor nodes, MBM defines a virtual market where nodes sell "actions" (such as taking a sensor reading or aggregating data) in response to global price information. Nodes take actions to maximize their own utility, subject to energy budget constraints. The behavior of the network is determined by adjusting the price vectors for each action, rather than by directly specifying local node actions, resulting in a globally efficient allocation of network resources. We present the market-based macroprogramming paradigm, as well as several experiments demonstrating its value for a sensor network vehicle tracking application.

1 Introduction

Sensor networks are an exciting new research area with novel applications in habitat monitoring [1, 13], medical care [20], and target tracking and pursuit [21]. The limited computation, communication, and energy available on individual sensor nodes presents a difficult programming challenge: that of decomposing complex, systemwide behaviors into the local actions to be taken at each node. While recent work has addressed aspects of this problem through high-level query interfaces [11] and communication layers [6, 19], the general problem of achieving globally efficient behavior from a distributed sensor network under changing environmental conditions is still wide open.

In this paper, we present a novel approach to global programming of entire sensor networks, based on the paradigm of *market-based macroprogramming (MBM)*. In this approach, individual sensor nodes act as self-interested agents that operate in a virtual market, and receive profit for performing simple, local actions in response to globally-advertised price information. Sensor nodes run a very simple cost-evaluation function, and global behavior is induced throughout the network by advertising price information that drives nodes to react. The prices can be dynamically tuned by the centralized market maker to meet systemwide goals of lifetime, accuracy, or latency based on the needs of the sensor network programmer. The "macroprogram" is therefore encoded in the process used to update price information in response to changing network conditions.

Consider distributed vehicle tracking, in which nodes must take periodic sensor readings, combine readings with other nearby nodes (for example, to compute the centroid of readings above some threshold), and report aggregated values to a base station. The periodicity with which each node performs sensing, computation, and communication has a deep impact on the overall lifetime of the network and the accuracy of the tracking system. The system must also tune these rates in response to changes in the target vehicle's location and velocity. Currently, programmers must implement these distributed algorithms by hand, and few tools exist to determine whether a given application will meet its energy and accuracy targets. In addition, retasking sensor networks is very difficult, as reprogramming sensor nodes over the air is generally prohibitive for energy and bandwidth reasons [8].

Market-based programming addresses many of these concerns, by providing a high-level programming environment in which the system can achieve globally efficient behavior under dynamic conditions. Nodes operate with naïveté of the global task and perform extremely simple local actions. Network retasking is accomplished through adjusting price vectors, rather than pushing new code to sensor nodes. This paper explores the use of market-based macroprogramming and presents initial experiments demonstrating its feasibility for distributed vehicle tracking. Of course, there are many theoretical and practical challenges to address with this approach, several of which are discussed herein. We expect that MBM can eventually be integrated with more conventional sensor network programming models, such as query languages [11] and virtual machines [8].

2 Background

The complexities of programming distributed sensor networks have led to some initial work on high-level programming interfaces and languages. These include communication abstractions such as directed diffusion [6, 3], abstract regions [19], Hoods [21], and GHT [17]. These systems make a good foundation for *macroprogramming*, which we define as the ability to program the sensor network as a whole. However, these previous approaches still require the system designer to explicitly program individual nodes' actions.

TinyDB [11], Cougar [22], and IrisNet [16] take a query-language-based view of macroprogramming by providing a high-level SQL or XML-based query interface to sensor network data. Queries are deployed into the network, streaming results to one or more base stations, and aggregation is used to

reduce communication overhead. However, capturing the sensor network at this level makes it difficult to implement specific behavior at a lower level than the query interface.

To date, little of this work has focused on allowing an application designer to meet systemwide goals such as latency or energy targets. Abstract regions [19] allow the programmer to tune the communication layer to trade off energy for accuracy, while TinyDB provides a *lifetime* keyword that scales the query sampling and transmission period to meet a user-supplied network lifetime [12]. In contrast, the goal of market-based macroprogramming is to achieve an efficient allocation of network resources to optimize specific objectives, such as data fidelity, subject to lifetime constraints. In MBM, this is accomplished by setting prices for virtual "goods" that consume energy and produce data, yielding a far more general structure than the periodic sampling and local communication in query interfaces such as TinyDB.

The MBM approach is inspired by Wellman's work on market-oriented programming for decentralized optimization [18, 15]. Under this paradigm, distributed computation and resource allocation are modeled as a market, and the interaction between agents is achieved through the buying and selling of goods and services. Market-based control has also been proposed in other application contexts [2]. However, we believe this paper is the first serious attempt to use market-oriented programming ideas to program a real distributed system.

Applying economic theory to the design of distributed systems yields new approaches to the complex process of allocating network-wide resources and defining global behavior. An allocation of resources is *Pareto optimal* if there is no reallocation of goods and services that can make a given agent better off without making another agent worse off. Pareto optimality yields a *globally efficient* allocation of sensor network resources. Such an allocation can be achieved by setting prices such that supply equals demand for each good and service. Such as system is said to be in *competitive equilibrium*. By the First Welfare Theorem [14], the competitive equilibrium allocation is guaranteed to be Pareto optimal under the classical assumption of monotone agent preferences.

MBM is attractive for programming sensor networks at the global level, as agents individually decide upon their actions given local knowledge. MBM also allows us to use the tools from general equilibrium theory to analyze the behavior and correctness of the system. For example, the benefit of adding (or removing) an agent can be determined by checking whether the agent will profit at current equilibrium prices. Prices provide aggregate feedback about the marginal systemwide value of a particular resource. Economic theory also allows the prediction of *market failure*, and provides a methodology to prevent failure through the enrichment of prices. Finally, market-based approaches extend to settings in which sensor networks are deployed across multiple domains and accessed by multiple users, perhaps with conflicting requirements.

3 Market-based Macroprogramming

Market-based macroprogramming (MBM) takes a resource allocation view of a sensor network, with nodes programmed to act as very simple economic agents. Prices coordinate the actions of agents, and are adjusted to induce useful global behavior. The market provides control over the behavior of sensor nodes through the propagation of price information.

Market-based macroprogramming requires a specification of the *actions* that each sensor node may take, as well as the local *utility function* that guides the decision-making of each node [18]. Examples of actions include sampling a sensor, aggregating data, and forwarding a message. The utility function represents the tradeoff between the profit for taking an action and the energy, bandwidth, or other resource cost associated with that action. Each *user* of the sensor network is also represented by an agent with an associated utility function, which captures the user's desire for the network to take certain actions. In this way, MBM allows multiple users to share the sensor network by offering differing payments for node actions. Each action has an associated *price* that may be adjusted at runtime to coordinate the actions of agents. Each node follows a very simple local algorithm to determine its behavior. Nodes continuously monitor prices and select actions to maximize their local utility, given the availability of local resources such as energy, sensor data, and pending messages to forward.

Energy is the critical resource in sensor networks, and allocating energy involves tradeoffs in terms of responsiveness, information quality, and lifetime. A well-functioning market should make *efficient* resource-allocation decisions, with a sequence of actions taken to maximize the total utility of all agents across time, subject to energy constraints.

3.1 MBM Primitives

MBM provides a set of primitives that collectively define the behavior of the system:

Goods and actions: The essential operation of the market in MBM is that agents perform *actions* to produce *goods* in return for payments. The actions that sensor nodes may take depend on the application, but typically include sampling a sensor, aggregating multiple sensor readings, or broadcasting a radio message. An action may be disallowed if the node does not currently have enough energy to perform the action. In addition, production of one good may have dependencies on the availability of others. For example, a node cannot aggregate sensor readings until it has acquired multiple readings.

Taking an action may or may not produce a good of value to the sensor network application. For example, listening for incoming radio messages is only valuable if a node hears a transmission from another node. Likewise, transmitting a sensor reading is only valuable if the reading has useful informational content. Therefore, not all actions will receive a payment. We assume that nodes can determine locally whether a given action deserves a payment. This works well for the simple actions considered here, although more complex actions (e.g., computing a function over a series of values) may require external notification for payments.

Currency: Markets need an internal currency to facilitate trade. The existence of currency facilitates multilateral trade through a sequence of bilateral trades and provides a common

Action	Energy consumed
sample (single sensor)	$1.637 \times 10^{-6} J$
send (single message)	$1.653 \times 10^{-3} J$
listen (for 1 sec)	$23.88 \times 10^{-3} J$
sleep (for 1 sec)	$90 \times 10^{-6} J$
aggregate (compute max of array)	$1.637 \times 10^{-6} J$

Figure 1: **Energy consumed for each sensor action, based on measurements of the Mica2 sensor node.**

scale for utility.

Agents: Agents are the basic abstraction in market-based systems, and are associated both with sensor nodes and users on the edge of the network. Agents have a utility function, and are provided with the autonomy to determine their own actions. By default, every sensor node follows simple price-taking behavior with respect to its local action space: continually selecting an action to maximize its profit, as long as that action is within its energy budget. *User agents* determine prices based on their preferences for nodes taking certain actions, and prices can be adjusted dynamically in response to changing network conditions. For example, a user agent may increase its preference for sensor readings when it believes a vehicle has entered the target tracking area.

Prices: The price on a good defines the payment that an agent can expect to receive for selling a unit of the good. Roughly, a price represents the current marginal (total) utility across all users of the sensor network for one unit of that good. The difference in prices on goods then provides feedback to induce useful local behavior across nodes.

Given a set of actions, goods produced by those actions, prices for each good, and energy cost for each action, each agent in an MBM system follows a very simple local algorithm to determine its behavior. A node simply monitors its local state and the global price vector, and selects the action that maximizes its utility function. Upon taking that action, the node's energy budget is reduced by the following amount, and the node may or may not receive a payment depending on whether its action produced a valuable good.

3.2 Application Example: Object Tracking

As a concrete example of an MBM-based application, we consider tracking a moving vehicle through a field of randomly-distributed sensors. Each sensor is equipped with a magnetometer capable of detecting local changes in magnetic field. The system-wide goal is to track the location of the moving vehicle as accurately as possible given a limit on the energy consumption of each node.

Each sensor agent has an initial endowment of energy and can take the following set of actions: *sample* a local sensor reading, *send* data towards the base station, *listen* for incoming messages, *sleep* for some interval, and *aggregate* multiple sensor readings (both local readings and those received over the radio). Aggregation is used to limit communication bandwidth and combines several readings into a single value. While a node is sleeping, it cannot receive radio messages, but consumes far less power than while listening. Figure 1 summarizes energy requirements for each action, based on measurements of the Mica2 sensor node. To reduce the amount of trace data generated by the simulation while still performing simulations of lengthy runs, we limited the rate at which nodes could take actions to 4 Hz.

3.2.1 Utility functions and parameters

We define the utility function for an action a to be:

$$u(a) = \begin{cases} \beta_a p_a & \text{if the action is } \textit{available} \\ 0 & \text{otherwise} \end{cases}$$

where a is the action under consideration, p_a is the current price for that action, and β_a is the *approximate probability of payment* for that action, which is learned by individual nodes as described below. An action may be *unavailable* if either the current energy budget is too low to take the action, or other dependencies have not been met (such as lack of sensor readings to aggregate), in which case the utility function evaluates to 0.

The utility function represents a node's *expected* profit over all possible actions. β_a is learned over time using an exponentially weighted moving average (EWMA) based on whether or not a node received payment for action a. Nodes are conditionally paid for the *listen* and *sample* actions based on whether or not a message was heard and whether or not a sample above a threshold was acquired, respectively; the β_a value for other actions is always 1. The expected profit may vary over time, so nodes must periodically explore by taking a putatively unprofitable action in case environmental changes have caused that action to become profitable. For example, if the vehicle comes into sensor range then *sample* actions will become profitable. We use an ϵ-greedy action selection policy. Most of the time a node chooses the action with the highest expected profit, but with probability ϵ the node will randomly choose an action from all available actions.

Learning β_a in this way allows nodes to automatically differentiate themselves over space and time to perform actions that are useful to the network as a whole. For example, a node close to the vehicle will take frequent sensor readings, whereas a node far from the vehicle but on the routing path between the sensors near the vehicle and the base station will listen for incoming radio messages to forward. Nodes adapt their behavior to changing conditions and do not need to be programmed explicitly to specialize in this way.

To capture the desire for nodes to consume energy at a regular rate, but still allow for occasional bursts of energy usage, we opt to use a token bucket model for a node's energy budget. Each node has a bucket of energy e that may hold no greater than some cap C of energy at any time. When a node takes an action, the appropriate amount of energy is deducted from the bucket. The bucket refills at a rate r that represents the average desired rate of energy usage. If a node cannot take any action because its bucket is too low, it must *sleep*, which places the node in the lowest-possible energy state.

3.3 Price selection and adjustment

In MBM, the global behavior of the network is controlled by the client establishing prices for each good. Prices are propagated to sensor nodes through an efficient global data dissemination algorithm, such as SPIN [4] or Trickle [10]. A user can also adjust

prices as the market runs in response to changes in the quality, rate, or latency of the data being produced by the network.

There is a complex relationship between prices and agent behavior. Raising the price for a good will not necessarily induce more nodes to produce that good; the dynamics of maximizing expected profits may temper a node's desire to take a given action. Given the complexities of agent operations and unknown environmental conditions, analytically solving for prices to obtain a desired result is not generally tenable. A better approach is to determine prices empirically based on an observation of the network's behavior at different price points. Prices can be readily tuned after deployment, since broadcasting a new price vector to an active network is not expensive.

4 Experiments

To demonstrate the efficacy of our market-based vehicle tracking system, we conducted several simulated experiments running under TOSSIM [9], a scalable sensor network simulator that runs actual TinyOS [5] applications. TOSSIM incorporates a realistic radio model based on traces of packet loss statistics in an actual sensor network. We simulated a network of 100 nodes distributed semi-irregularly in a 100x100 meter area. The base station that is collecting sensor readings in located in the upper-left corner of this area. Nodes communicate to the base station through GPSR [7], a geographic routing protocol that we have implemented in the TinyOS environment [19].

4.1 Individual node behavior

To illustrate the behavior of a particular node in detail, Figure 2 depicts the actions taken, energy budget, and β values for node 23, which is along the path of the vehicle in the sensor network. As the vehicle approaches along its circular path at time $t = 350$, the node determines that it will be paid to sample, aggregate, and send sensor readings. As the vehicle departs around time $t = 500$, the node returns to its original behavior. At other times (e.g., $t = 520$), the node receives messages from other nodes and routes them towards the base station, explaining the sudden spikes in β for the listen action.

Also note that the node performs a large number of listen and sample actions even when its utility for doing so is low (even zero). This is because the node has enough energy to perform these actions, and the ϵ-greedy action selection policy dictates that it will explore among these alternatives despite negligible utility.

4.2 Energy budget and price adjustment

MBM offers a number of tunable design parameters to an application builder. For sake of brevity, we focus on two: the prices of the various goods and the per-diem energy budget allocated to each node. Recall that nodes maximize their utility by taking actions subject to an energy budget, and that the prices represent the application's preferences for taking certain actions over others. We consider two energy budgets: a "low" budget of 400 J/day (permitting a lifetime of 77 days[1]), and a "high" budget of 3000 J/day (lifetime 10 days). These values were chosen

[1]We assume the device runs at 3V and that 2 AA batteries provide 2850 mA-hours of capacity.

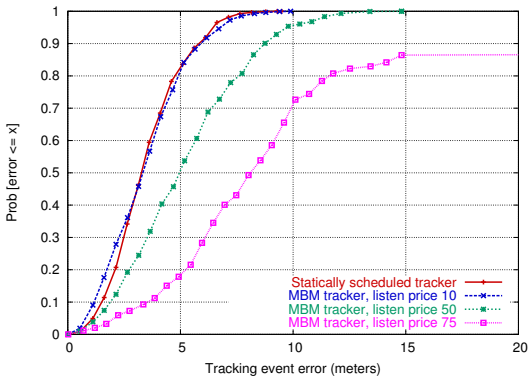


Figure 3: **CDF of tracking error as the *listen* price is adjusted.**

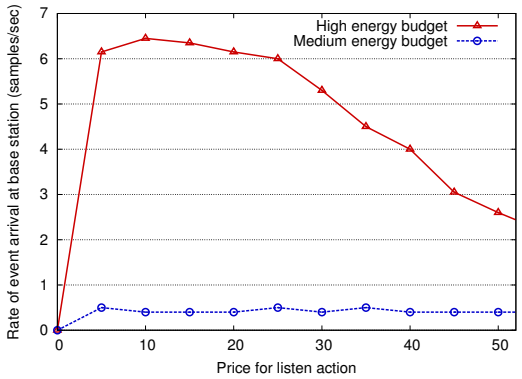


Figure 4: **Rate of tracking event arrival at base station, as the *listen* price is adjusted.**

to correspond to reasonable values for a battery-powered sensor network. Note that in the case of the "high" energy budget, a node's choice of action is unconstrained by energy considerations.

Figure 3 shows the accuracy of the vehicle tracker, with the high energy budget, as the price for the *listen* action is adjusted. For comparison, we show data for a simpler version of the tracker where every node uses a static schedule for selecting actions. In the statically-scheduled version, every node in the network periodically samples, aggregates, transmits, and listens for incoming messages according to a rate that allows the node to operate within its daily energy budget. This is the traditional way of scheduling node operations, typified by the *query epoch* in TinyDB [11].

As the figure shows, increasing the price for the *listen* action degrades the tracker's accuracy, since fewer nodes are taking samples. While the statically-scheduled tracker performs as well as the MBM tracker for a listen price of 10, every node in the statically-scheduled network is regularly sampling, aggregating, and transmitting samples regardless of its position in the network and the location of the target vehicle. In the MBM version, only nodes near the vehicle are induced to take samples and transmit data at all. It is important to keep in mind that this complex behavior was not programmed manually; it is driven entirely by the nodes' utility functions and the process of learning β for each action.

Figure 4 shows the number of position estimates generated by

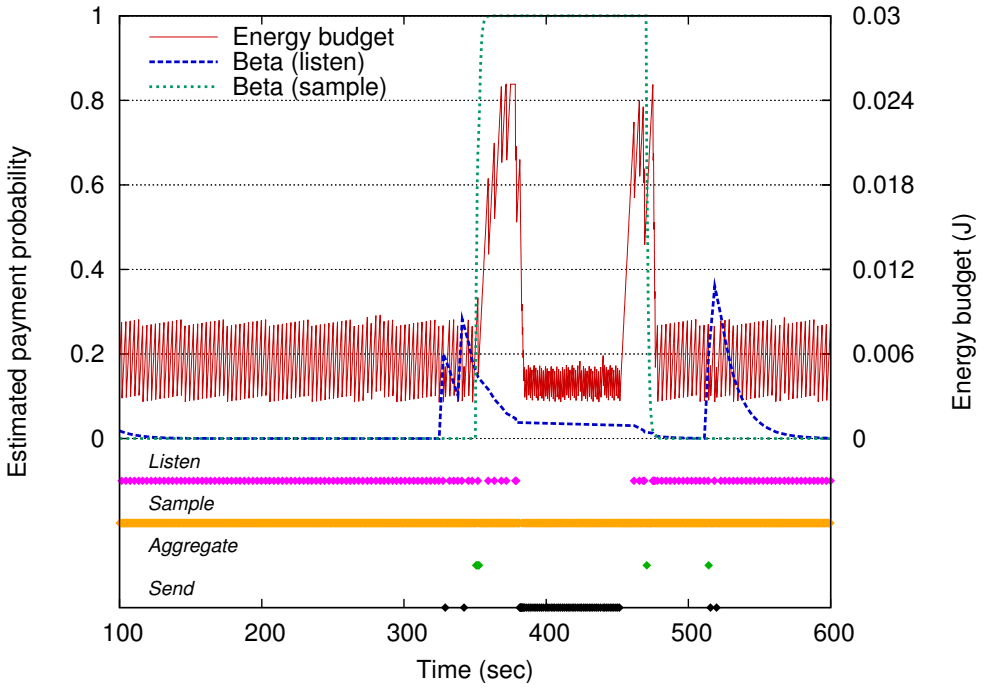
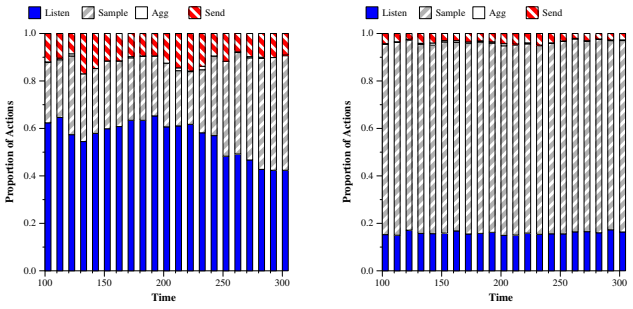


Figure 2: **Actions and energy budget for a single node.** *This figure shows the actions taken, the energy budget, and the β values for the listen and sample actions for node 23, which is along the path of the vehicle in the sensor network.*



(a) Large energy budget **(b) Small energy budget**

Figure 5: **Exploiting heterogeneous energy budgets.** *Here, 20% of the nodes are given a large energy budget of 3000 J/day (a), where the rest of the nodes use the standard energy budget of 400 J/day (b). The large energy budget nodes automatically take on a greater proportion of the energy load in the system, choosing to perform a far greater number of listen and send actions than the low-energy nodes.*

the network as the price for the *listen* action is increased, for both energy budgets. There is an ideal setting between "listening too little" and "listening too much," where the number of readings delivered by the network is maximized. With MBM, nodes near the vehicle learn that sampling is profitable and acquire a larger number of samples during the times that the vehicle is nearby. Likewise, nodes along the routing path from sensors near the vehicle to the base station learn that listening is profitable and spend a greater amount of time receiving data to be forwarded.

MBM also allows nodes to be differentiated with respect to their energy budgets. For example, certain nodes may have access to a large power supply and can perform a different set of operations than nodes operating off of small batteries. Likewise, advertising different price vectors to different nodes allows them to be customized to take certain actions.

Figure 5 shows the behavior of the tracking network where 80% of the nodes operate with the low energy budget and 20% of the nodes operate with the large energy budget. The large energy budget nodes automatically elect to perform a greater number of listen and send actions, while the other nodes mostly perform sample actions, which consume far less energy overall. Identical prices are used throughout the network, showing that differences in energy budget have profound effect on the self-scheduling of nodes.

5 Future Work and Conclusions

Market-based macroprogramming brings a number of new ideas to the design of sensor network applications. Exploiting techniques from economic theory and market design should yield new insights into the construction of robust, efficient distributed systems with severe resource constraints. Our initial work on MBM raises a number of interesting questions that we wish to explore in future work. These are described in summary below.

Richer pricing: More complex pricing schemes can be used to induce sophisticated behaviors in the network. For example, rather than pricing only those goods that result from atomic actions, we can price *sequences* of actions. Consider aggregating multiple sensor readings into a single value for transmission. Rather than price the final aggregate value and requiring an agent to reason about a sequence of actions to achieve that result, we can establish prices for each step in the sequence and introduce control or data dependencies between actions. Another question is that of *location-based prices*, in which goods are priced

differently in different areas of the network. This can be used to cause certain nodes to act primarily as local message routers, while other nodes devote themselves to local sensing.

Equilibrium prices: We have experimented with dynamically adjusting prices to meet demands set by the base station. Unfortunately, for our tracking application the supply of goods that the network can provide varies greatly over time, even when prices are held constant. This is because supply is dependent on factors other than prices, making it difficult to use prices alone to meet equilibrium conditions. For example, there is a large supply of readings available when the vehicle is close to the base station, but a far smaller supply when the vehicle is on the edge of the network far from the base station. We plan to investigate alternative pricing models that can capture these complexities in a market-oriented context.

Multiple users: Market-based techniques are well-suited for addressing issues that arise when multiple competing users wish to share a sensor network. Each user advertises its demand for taking certain actions on its behalf, and equilibrium prices are calculated to maximize the utility of all agents (both sensor nodes and users) in the system. The competitive equilibrium condition ensures that the resulting allocation is Pareto optimal. We believe this is an important advantage of market-based techniques and we intend to explore this angle in future work.

Reinforcement learning: One of the more interesting aspects of our system is the ability of nodes to adapt their operation over time by estimating the *expected profit* of actions. Our scheme for learning β values is a simple form of reinforcement learning. Reinforcement learning provides a large body of techniques that can be applied to multistage decision problems for which there is no (known) good model of the system. An interesting path for future work is the integration of more sophisticated reinforcement learning techniques for sensor network macroprogramming.

We believe that market-based programming can be applied to the design of complex distributed systems in general, not just sensor networks. In situations where multiple users are competing for a shared infrastructure, or multiple competing services are running within a network, MBM allows one to rely on the economic principles of rationality and self-interest to reason about the behavior of the system. We are planning further experiments with MBM to understand its potential for distributed programming.

References

[1] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proc. the Workshop on Data Communications in Latin America and the Caribbean*, Apr. 2001.

[2] S. H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation.* World Scientific, 1996.

[3] J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proc. the 18th SOSP*, Banff, Canada, October 2001.

[4] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. the 5th ACM/IEEE Mobicom Conference*, August 1999.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Proc. the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, Boston, MA, USA, Nov. 2000.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. International Conference on Mobile Computing and Networking*, Aug. 2000.

[7] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, August 2000.

[8] P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. In *Proc. the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*, 2002.

[9] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proc. the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[10] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proc. the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, March 2004.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. the 5th OSDI*, December 2002.

[12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. the ACM SIGMOD 2003 Conference*, June 2003.

[13] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, USA, Sept. 2002.

[14] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[15] T. Mullen and M. P. Wellman. Some issues in the design of market-oriented agents. In W. et al., editor, *Intelligent Agents: Theories, Architectures and Languages*, volume 2. Springer-Verlag, 1996.

[16] S. Nath, Y. Ke, P. B. Gibbons, B. Karp, and S. Seshan. IrisNet: An architecture for enabling sensor-enriched Internet service. Technical Report IRP-TR-03-04, Intel Research Pittsburgh, June 2003.

[17] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage in sensornets. In *Proc. the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, September 2002.

[18] M. P. Wellman. Market-oriented programming: Some early lessons. In S. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[19] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proc. the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, March 2004.

[20] M. Welsh, D. Myung, M. Gaynor, and S. Moulton. Resuscitation monitoring with a wireless sensor network. In *Supplement to Circulation: Journal of the American Heart Association*, October 28, 2003.

[21] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: A neighborhood abstraction for sensor networks. In *Proc. the International Conference on Mobile Systems, Applications, and Services (MOBISYS '04)*, June 2004.

[22] Y. Yao and J. E. Gehrke. The Cougar approach to in-network query processing in sensor networks. *ACM Sigmod Record*, 31(3), September 2002.